Database Manager

Return Types

JSFoundSet JSRecord

Method Summary

String	getDataSource(serverName, tableName) Returns the datasource corresponding to the given server/table.
String	getDataSourceServerName(dataSource) Returns the server name from the datasource, or null if not a database datasource.
String	getDataSourceTableName(dataSource) Returns the table name from the datasource, or null if the specified argument is not a database datasource.
JSFoundSet	getFoundSet(dataSource) Returns a foundset object for a specified datasource or server and tablename.
Boolean	hasRecords(foundset) Returns true if the (related)foundset exists and has records.
Boolean	hasRecords(record, relationString) Returns true if the (related)foundset exists and has records.
Boolean	saveData() Saves all outstanding (unsaved) data and exits the current record.

Method Details

getDataSource

String **getDataSource** (serverName, tableName)
Returns the datasource corresponding to the given server/table.
Parameters

{String} serverName - The name of the table's server.
{String} tableName - The table's name.
Returns

String - The datasource of the given table/server.
Sample

```
var datasource = databaseManager.getDataSource('example_data',
'categories');
```

getDataSourceServerName

String **getDataSourceServerName** (dataSource)
Returns the server name from the datasource, or null if not a database datasource.
Parameters

{String} dataSource - The datasource string to get the server name from.
Returns

String - The servername of the datasource.
Sample

```
var servername = databaseManager.getDataSourceServerName(datasource);
```

getDataSourceTableName

String **getDataSourceTableName** (dataSource)
Returns the table name from the datasource, or null if the specified argument is not a database datasource.
Parameters

{String} dataSource - The datasource string to get the tablename from.
Returns

String - The tablename of the datasource.
Sample

```
var theTableName = databaseManager.getDataSourceTableName(datasource);
```

getFoundSet

JSFoundSet **getFoundSet** (dataSource)
Returns a foundset object for a specified datasource or server and tablename.
Parameters

{String} dataSource - The datasource to get a JSFoundset for.
Returns

JSFoundSet - A new JSFoundset for that datasource.
Sample

```
// type the foundset returned from the call with JSDoc, fill in the
right server/tablename
/** @type {JSFoundset<db:/servername/tablename>} */
var fs = databaseManager.getFoundSet(controller.getDataSource())
var ridx = fs.newRecord()
var record = fs.getRecord(ridx)
record.emp_name = 'John'
databaseManager.saveData()
```

hasRecords

Boolean **hasRecords** (foundset)
Returns true if the (related)foundset exists and has records.
Parameters

{JSFoundSet} foundset - A JSFoundset to test.
Returns

Boolean - true if the foundset/relation has records.
Sample

```
if (elements.customer_id.hasRecords(orders_to_orderitems))
{
        //do work on relatedFoundSet
}
//if (elements.customer_id.hasRecords(foundset.getSelectedRecord(),
'orders_to_orderitems.orderitems_to_products'))
//{
//        //do work on deeper relatedFoundSet
//}
```

hasRecords

Boolean **hasRecords** (record, relationString)
Returns true if the (related)foundset exists and has records.

Parameters

{JSRecord} record - A JSRecord to test.
{String} relationString - The relation name.

Returns

Boolean - true if the foundset/relation has records.

Sample

```
if (elements.customer_id.hasRecords(orders_to_orderitems))
{
        //do work on relatedFoundSet
}
//if (elements.customer_id.hasRecords(foundset.getSelectedRecord(),
'orders_to_orderitems.orderitems_to_products'))
//{
//        //do work on deeper relatedFoundSet
//}
```

saveData

Boolean **saveData** ()
Saves all outstanding (unsaved) data and exits the current record.
Optionally, by specifying a record or foundset, can save a single record or all records from foundset instead of all the data.

NOTE: The fields focus may be lost in user interface in order to determine the edits.
SaveData called from table events (like afterRecordInsert) is only partially supported depeding on how first saveData (that triggers the event) is called.
If first saveData is called with no arguments, all saveData from table events are returning immediatelly with true value and records will be saved as part of first save.
If first saveData is called with record(s) as arguments, saveData from table event will try to save record(s) from arguments that are different than those in first call.
SaveData with no arguments inside table events will always return true without saving anything.

Returns

Boolean - true if the save was done without an error.

Sample

```
databaseManager.saveData();
//databaseManager.saveData(foundset.getRecord(1));//save specific record
//databaseManager.saveData(foundset);//save all records from foundset

// when creating many records in a loop do a batch save on an interval
as every 10 records (to save on memory and roundtrips)
// for (var recordIndex = 1; recordIndex <= 5000; recordIndex++)
// {
//                 foundset.newRecord();
//                 someColumn = recordIndex;
//                 anotherColumn = "Index is: " + recordIndex;
//                 if (recordIndex % 10 == 0) databaseManager.saveData();
// }
```